Microsoft

Azure Resource Manager from A to Z

David Pazdera
Azure Cloud School meetup group

Speaker's bio

David Pazdera

Cloud Solution Architect, Microsoft

20+ years of experience (IT Pro)

infrastructure, identity, UCC, automation, ITSM

1 wife, 2 kids, no cats or dogs

@pazdedav

about.me/davidpazdera

www.linkedin.com/in/pazdedav/

Demo-rich session means…

Slides with key points, diagrams, references with resources, and quick demos on the way…

“Real life” scenarios and end-to-end examples on how they could be solved…

A journey about building Azure deployment maturity!

Agenda

Episode 1: Template for Jen

Simplifying resource deployment for users…

Episode 2: Douglas wants a catalogue

Building a simple Service Catalogue…

Episode 3: New CISO arrives

Adding some security and compliance stuff…

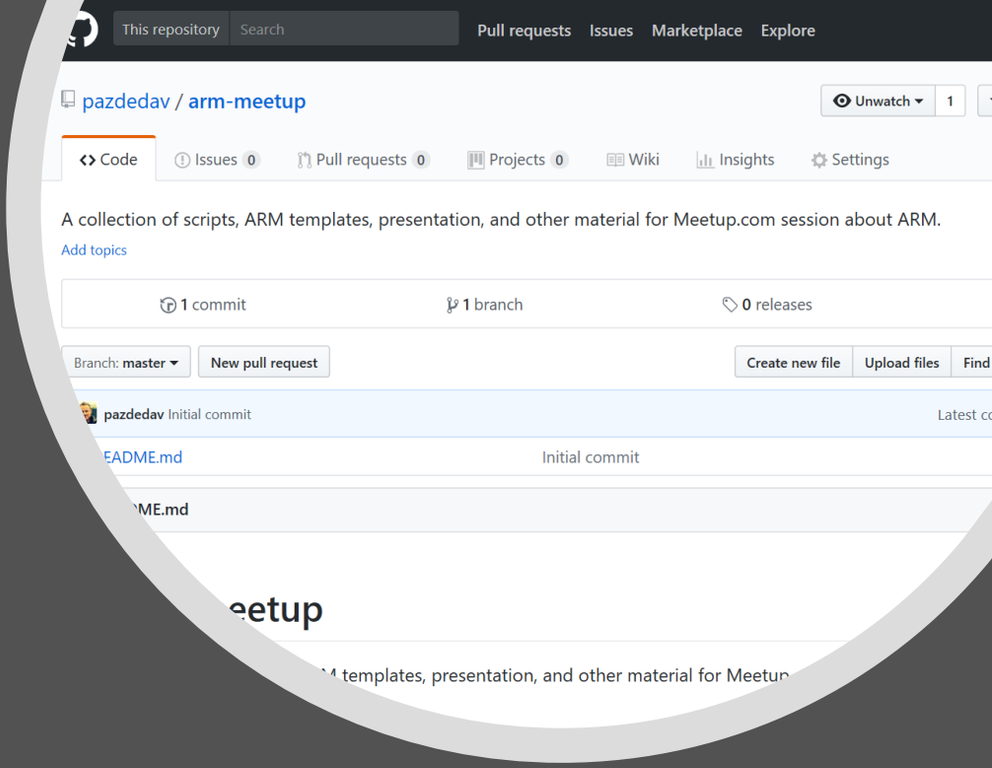
Episode 4: Automated Moss

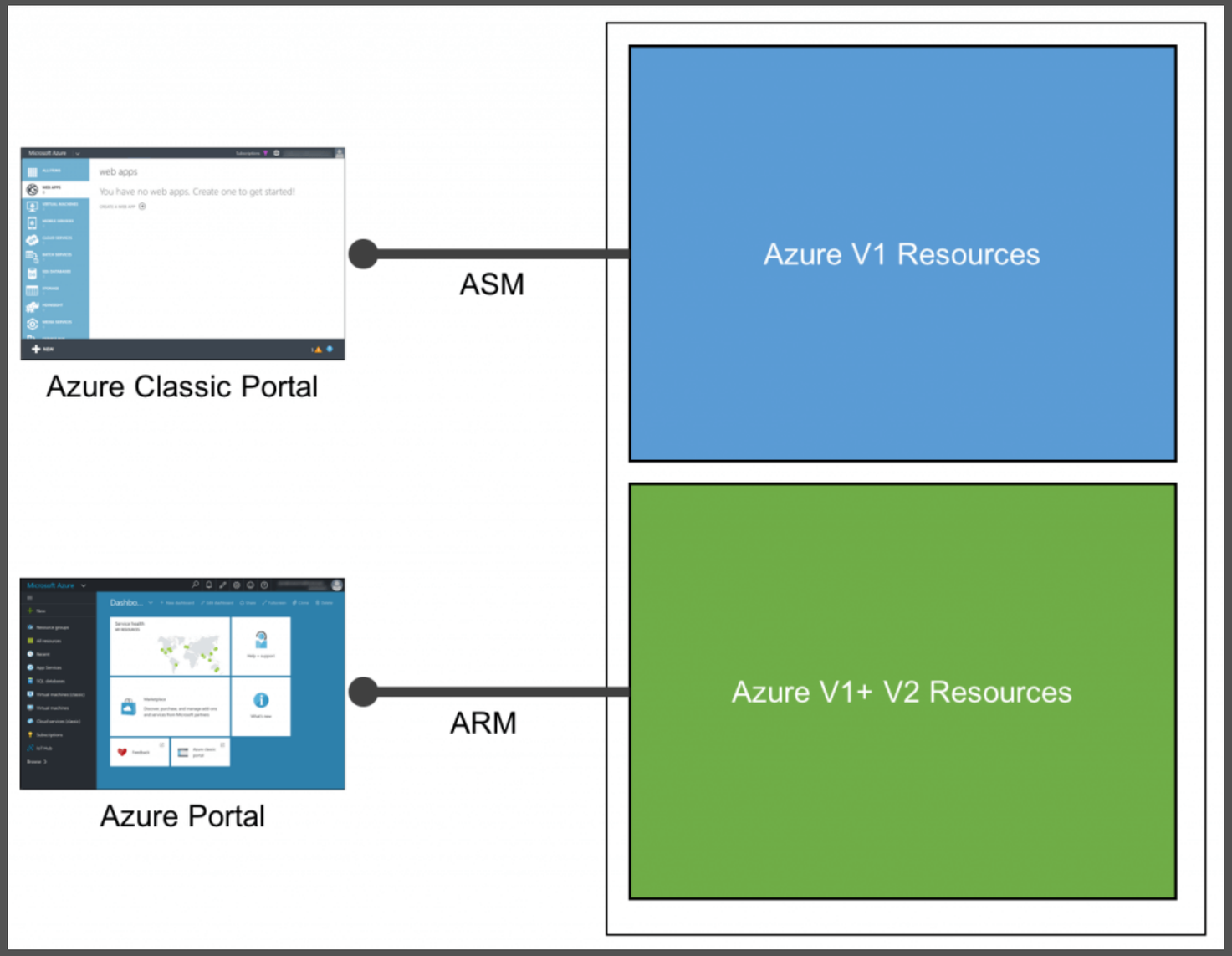
Automating deployments and IT processes…

Everything is on GitHub

github.com/pazdedav/arm-meetup

- Slides
- Templates
- Scripts
- Etc.





The ASM is dead, long live the ARM?

ARM Fundamentals

Azure Resource Manager

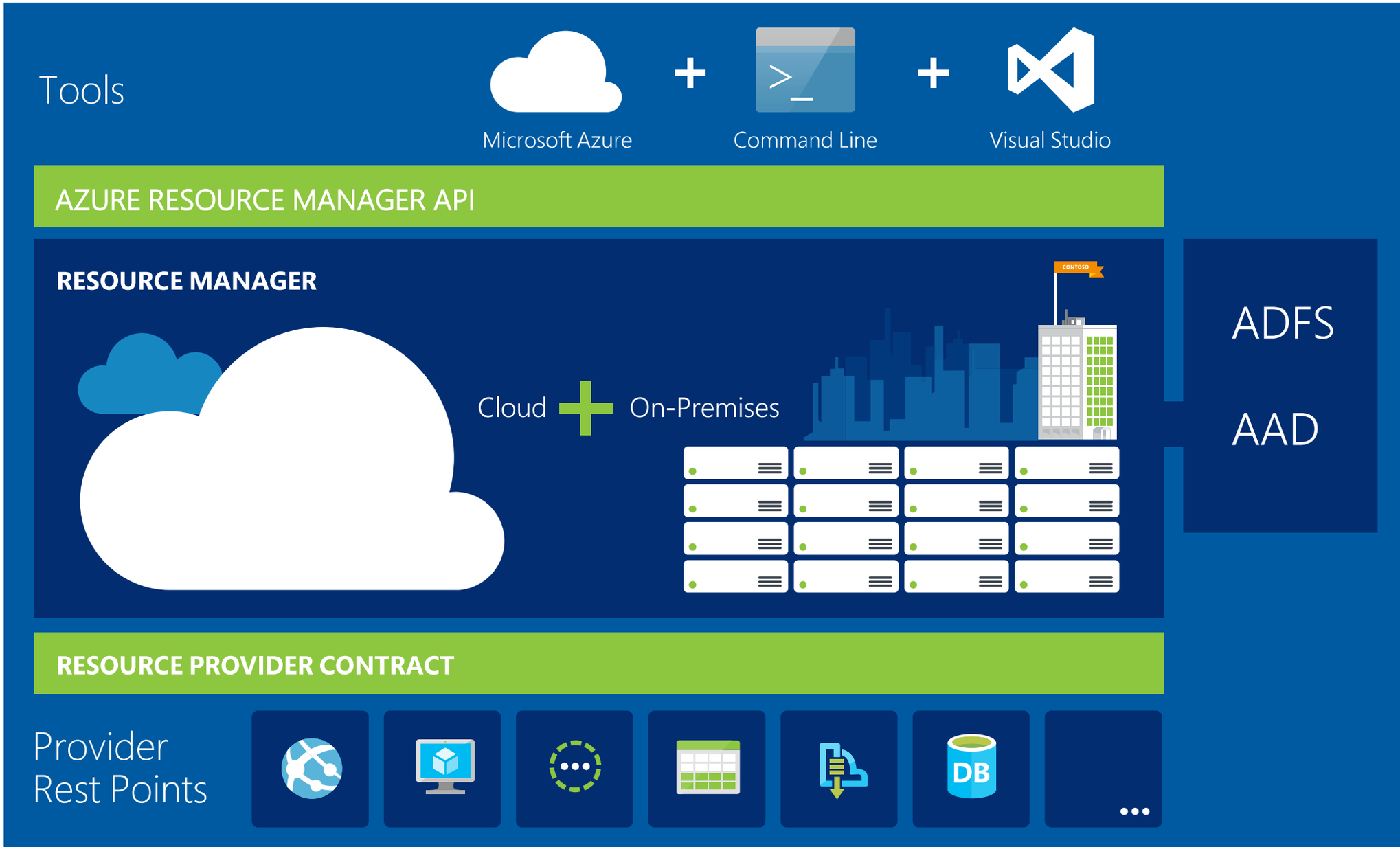
Benefits

- Desired-state deployment
- Faster deployments
- Role-based access control (RBAC)
- Resource-provider model
- Common interface for Azure and Azure Stack
- Centralized auditing, simple tagging and grouping

Deployments

- Template-driven
- Declarative → Infra as Code
- Idempotent
- Multi-service
- Multi-region
- Extensible

Consistent Management Layer



Resource Group – a unit of management

- Tightly coupled **container** of multiple resources, can span regions, not subscriptions
- Every resource ***must*** exist in one and only one RG
  - Add / remove a resource to a RG at any time
  - Possibility to move resources to different RGs / subscriptions (limitations)
- **ALM** – Deployment, update, delete and status
- **Declarative** solution for Deployment – "Config as Code"
- **Grouping** – Metering, billing, quota: applied and rolled up to the group
- Access Control – Scope for **RBAC** permissions

Resource Provider

- Used to deploy and manage specific types of resources
- Identified by provider namespace
  - e.g. `Microsoft.Compute, Microsoft.Storage, Microsoft.Web, NewRelic.APM`
- Resource types
  - Each provider namespace manages one or more resource types: e.g. `virtualMachines, loadBalancers`
- Different regional availability and apiVersions
- Portal: Subscriptions – Resource Providers blade
- PSH/CLI: `Get-AzureRmResourceProvider` or `az provider list`

Resources

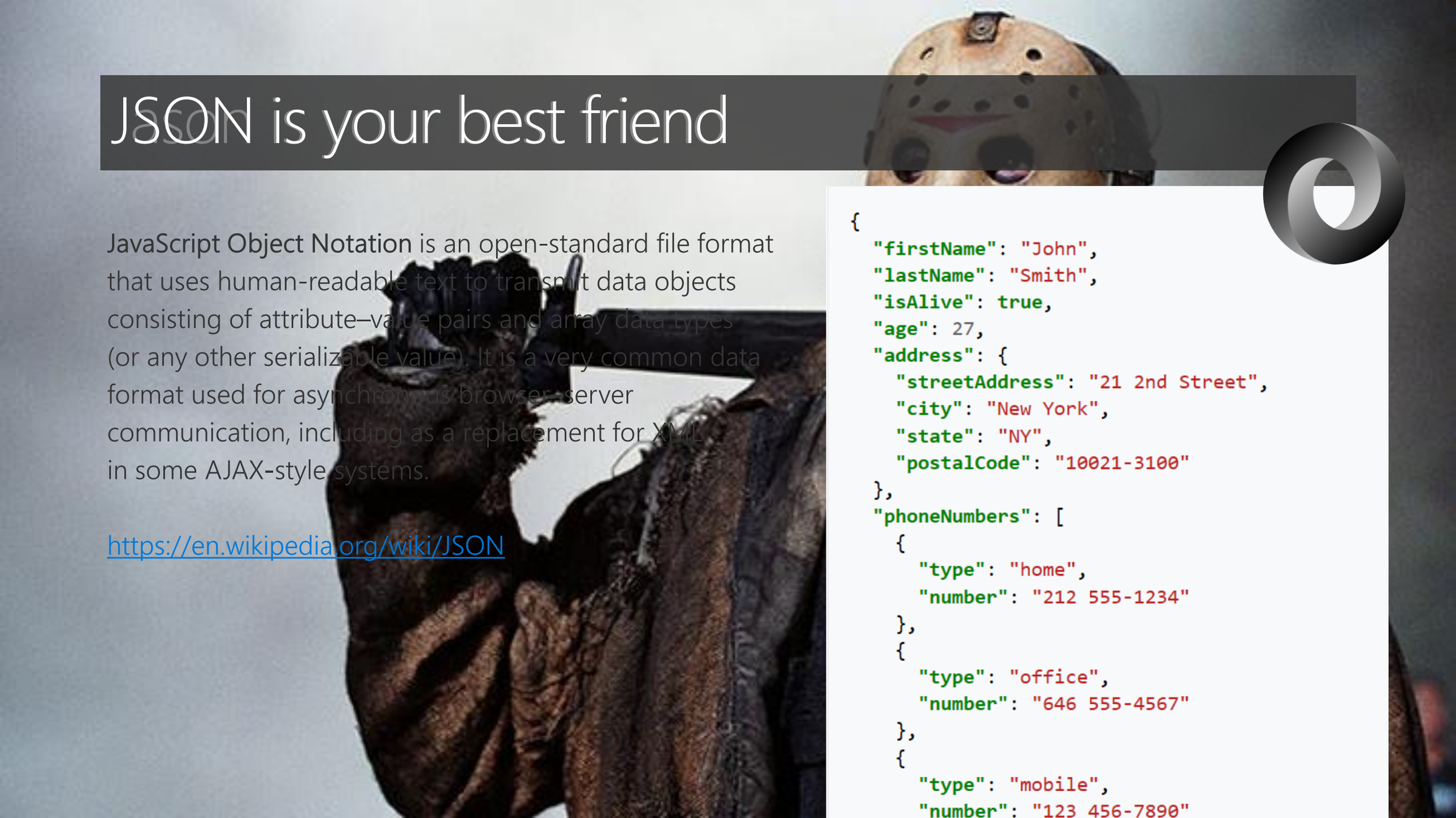
- Name - Unique for resource group and resource type
  - `Microsoft.Compute/virtualMachines`
- Id - Unique across Azure

  `/subscriptions/GUID/resourceGroups/myRG/providers/Microsoft.Compute/virtualMachines/vmName`
- Location
- ResourceType
- ResourceGroup
- Properties
  - Additional properties specific to the resource provider

JSON is your best friend

JavaScript Object Notation is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

https://en.wikipedia.org/wiki/JSON



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
```

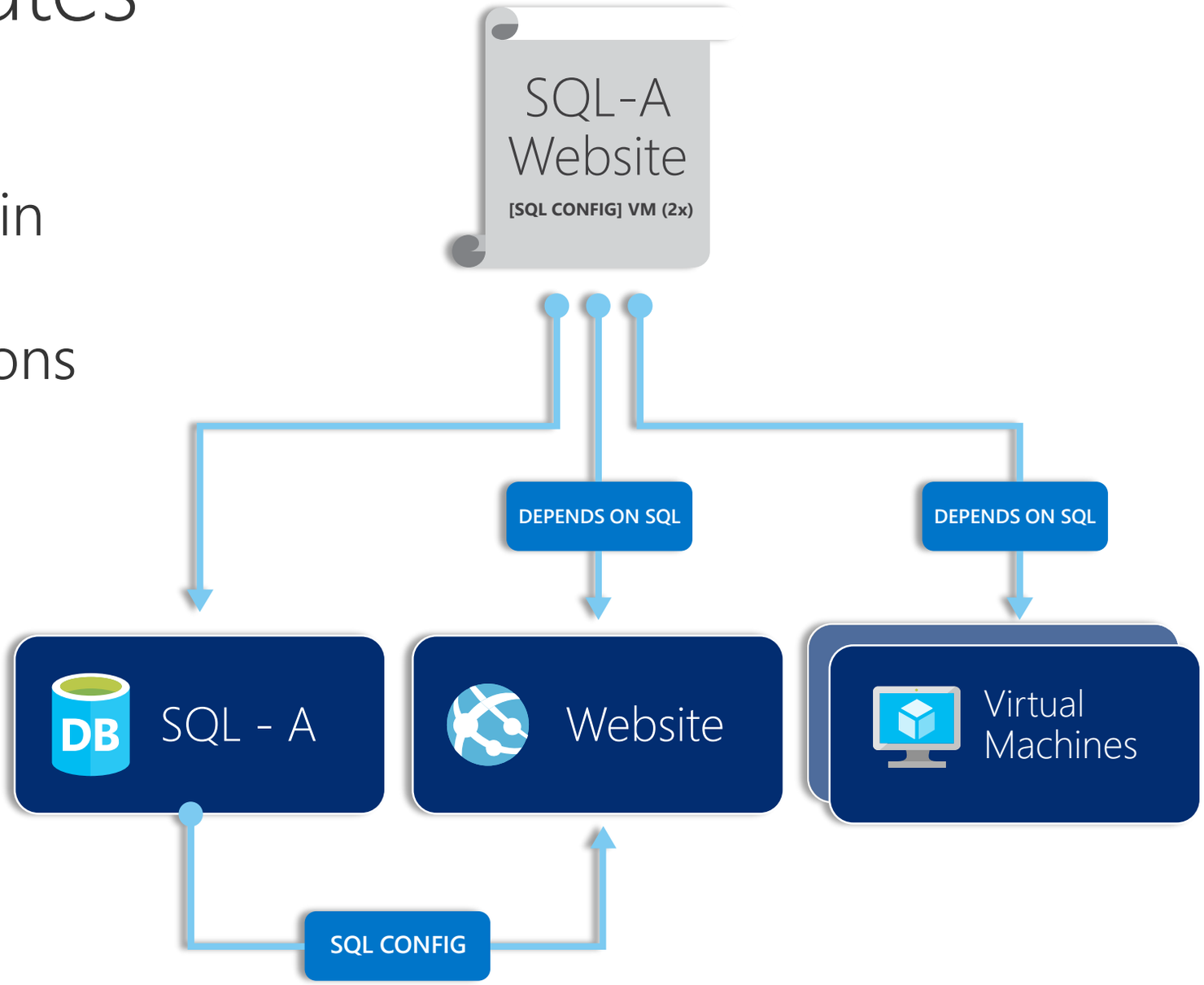
Deployment Templates

What?

- Source file, can be checked-in
- Specifies resources, dependencies and connections
- Parameterized input/output

Why?

- Simplify orchestration
- Provide cross-resource configuration and update support



Template Structure

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "",
    "parameters": {  },
    "variables": {  },
    "resources": [  ],
    "outputs": {  }
}
```

- **$schema:** Location of the JSON schema file that describes the version of the template language.
- **contentVersion:** Version of the template (such as 1.0.0.0). Used for versioning.
- **parameters:** Values that are provided when deployment is executed to customize resource deployment.
- **variables:** Values that are used as JSON fragments in the template to simplify template language expressions.
- **resources:** Types of services deployed. Minimum of 1.
- **output:** Values that are returned after deployment.

Note: Yellow sections are mandatory.

Parameters

```
"parameters": {
   "<parameterName>" : {
     "type" : "<type-of-parameter-value>",
     "defaultValue": "<optional-default-value-of-parameter>",
     "allowedValues": [ "<optional-array-of-allowed-values>" ],
     "minValue": <optional-minimum-value-for-int-parameters>,
     "maxValue": <optional-maximum-value-for-int-parameters>,
     "minLength": <optional-minimum-length-for-string-secureString-array-parameters>,
     "maxLength": <optional-maximum-length-for-string-secureString-array-parameters>,
     "metadata": {
         "description": "<optional-description-of-the parameter>"
     }
   }
}
```

- Parameter name and type are required
- Allowed JSON types: string, secureString, int, bool, object, secureObject, array

Variables and Parameters

```
"variables": {
   "<variable-name>": "<variable-value>",
   "<variable-name>": {
       <variable-complex-type-value>
   }
}
```

```
"parameters": {
   "username": {
     "type": "string"
   },
   "password": {
     "type": "secureString"
   }
 },
 "variables": {
   "connectionString": "[concat('Name=', parameters('username'), ';Password=', parameters('password'))]"
}
```

Resources

```
"resources": [
  {
    "apiVersion": "<api-version-of-resource>",
    "type": "<resource-provider-namespace/resource-type-name>",
    "name": "<name-of-the-resource>",
    "location": "<location-of-resource>",
    "tags": "<name-value-pairs-for-resource-tagging>",
    "comments": "<your-reference-notes>",
    "dependsOn": [
      "<array-of-related-resource-names>"
    ],
    "properties": "<settings-for-the-resource>",
    "resources": [
      "<array-of-dependent-resources>"
    ]
  }
]
```

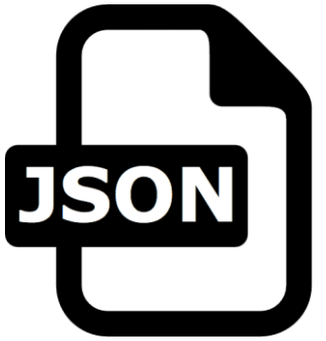
- Required:
  - apiVersion
  - type
  - name

Output

```
"outputs": {
    "<outputName>" : {
      "type" : "<type-of-output-value>",
      "value": "<output-value-expression>",
    }
}
```

- Name required
- Output types supported are same as for parameters

Additional information



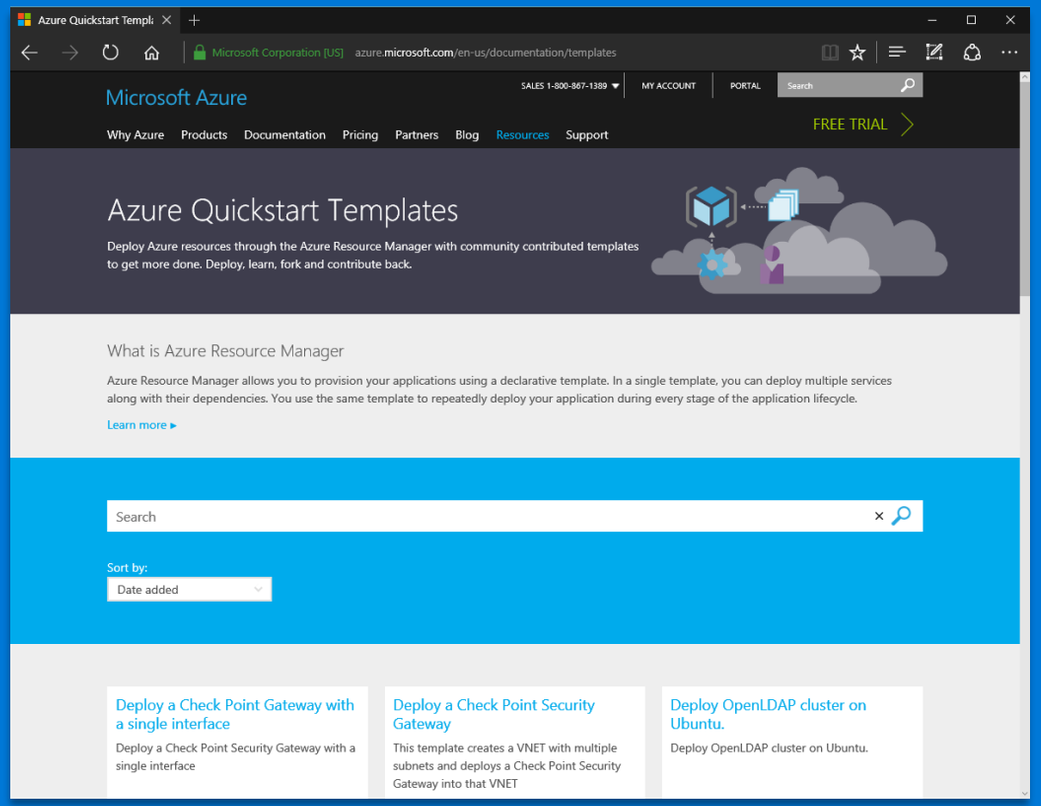
Useful tools

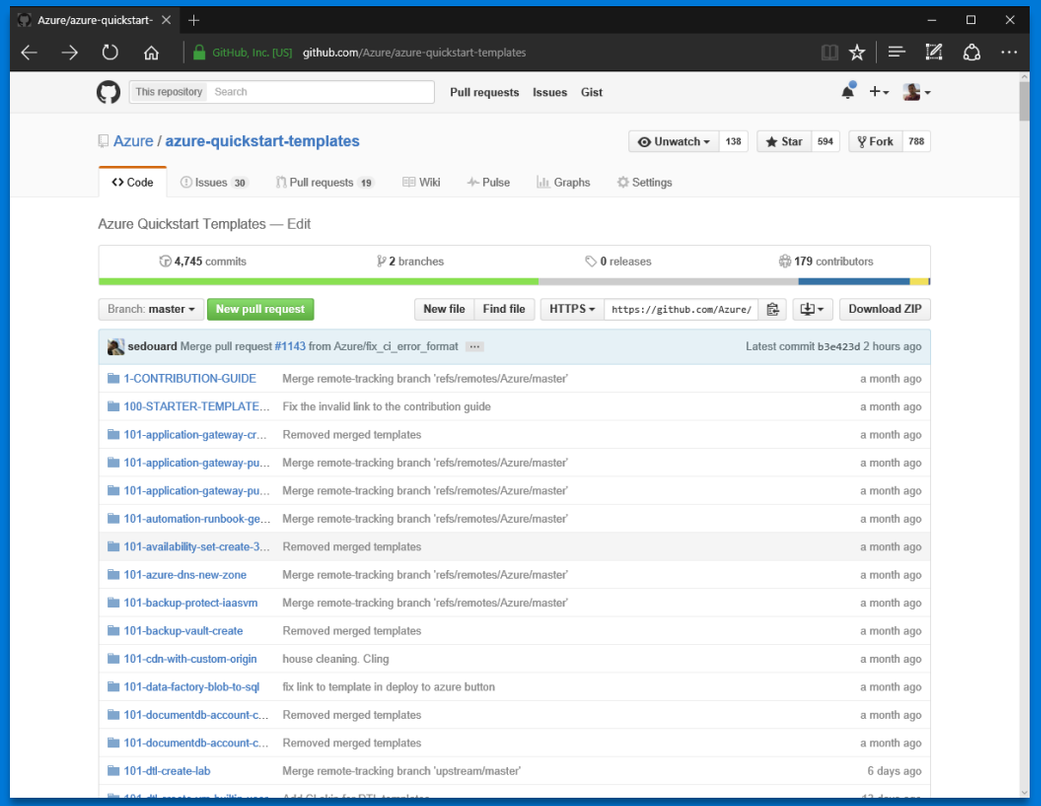
- JSON Validator - https://jsonlint.com/
- JSON Formatter - https://jsonformatter.org/

Limits

- File size: 1 MB for template, 64 KB for parameters file
- Content: 256 parameters, 256 variables, 800 resources (including copy count), 64 output values, 24,576 characters in a template expression
- Workarounds:
  - nested templates
  - Combining values into objects
- More info here

QuickStart Templates





*Azure.com**

https://azure.microsoft.com/en-us/documentation/templates/

GitHub

https://github.com/Azure/azure-quickstart-templates

**Azure.com->Resources->Templates*

Guidance

- Define and deploy your infrastructure through the **declarative syntax** in Resource Manager templates, rather than through imperative commands.
- Define **all deployment and configuration steps in the template**. You should have no manual steps for setting up your solution.
- Run **imperative commands to manage your resources**, such as to start or stop an app or machine.
- Arrange resources with the **same lifecycle** in a resource group. Use tags for all other organizing of resources.

Episode 1: Template for Jen

Simplifying resource deployment for users…

Scenario

- Jen wants to have a convenient way to get a new VM without typing all that stuff in the portal. Roy is tasked to make something good…
- In this episode:
  - Template export (3 options)
  - Template authoring in VS Code
  - Extensions for ARM in VS Code
  - Template publishing and sharing for Jen

VS Code extension for ARM

Icon	Extension Name	Description
	Azure Account	Provides a single Azure sign-in and subscription filtering. Makes Azure's Cloud Shell available in VS Code's integrated terminal.
	Azure Resource Manager Tools	Provides language support for ARM deployment templates and template language expressions.
	Azure Tools for Visual Studio Code	Convenient features for devs: template repository search, deployment within VS Code, template exports, etc.
	Azure CLI Tools	Tools for developing and running commands of the Azure CLI. Gives IntelliSense and snippets for .azcli scrapbooks.
	Azure Extension Pack	A collection of extensions for working with Azure resources in VS Code, e.g. App Services, Functions, Microservices (docker tools, AKS, ACR), Storage, Databases, VSTS, IoT
	Azure ARM Template Helper	VS-like tree-view for ARM templates including a few helpers (Preview)

Note: Extension icons contain links to VS Marketplace.

ARM API and deployments

API first strategy

- REST API
  - Postman, cURL, ARMClient
- Portal
  - GUI + Azure Cloud Shell
- PowerShell
  - Azure cmdlets + Invoke-RestMethod
- CLI 2.0 (cross-platform)
- Client libraries (SDKs)
  - Libraries for .NET, Java, Python, Ruby, Node.js

```
https://management.azure.com/subscriptions/{{subscriptionId}}/resourcegroups?api-version=2017-05-10
```


```
Get-AzureRmResourceGroup
```

```
az group list [--tag]
```

Get the Azure libraries for .NET
Get started
API reference

Get the Azure libraries for
Get started
API reference

Get the Azure libraries for Java
Get started
API reference

Get the Ruby SDK
Get started
API reference
Get samples
Learn more

ARM conversion to RP operations

JSON

```
"resources": [
  {
    "apiVersion": "2016-01-01",
    "type": "Microsoft.Storage/storageAccounts",
    "name": "mystorageaccount",
    "location": "westus",
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {
    }
  }
]
```

HTTP

PUT

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Storage/storageAccounts/mystorageaccount?api-version=2016-01-01
```

REQUEST BODY

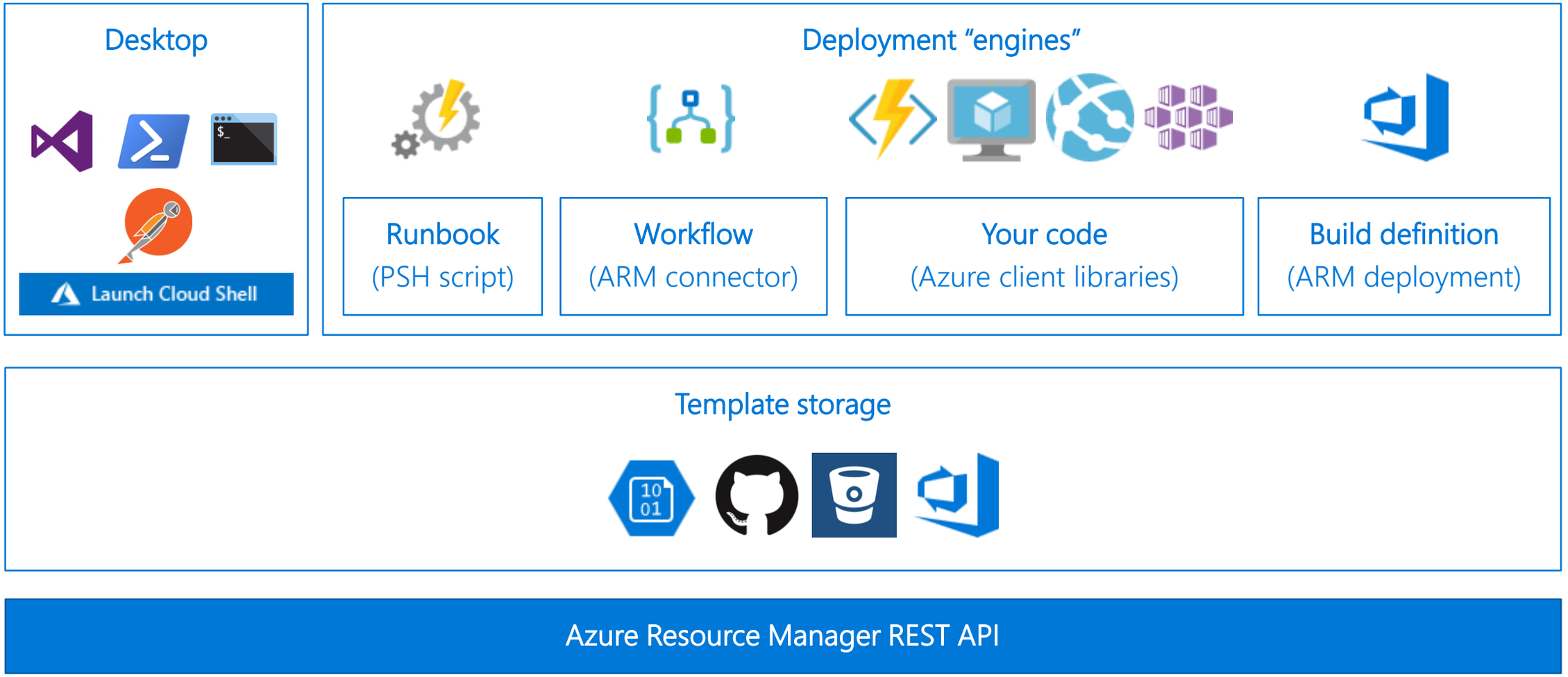
```
{
  "location": "westus",
  "properties": {
  }
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "Storage"
}
```

Additional facts about ARM

- Resource Manager is supported in all regions
- Resources you deploy might not – Azure Status
- Limitations on your subscription that prevent you from using some regions that support the resource (e.g. policy)

PRODUCTS AND SERVICES	NON-REGIONAL*	NORTH EUROPE	WEST EUROPE	UK WEST	UK SOUTH	GERMANY NON-REGIONAL	GERMANY CENTRAL	GERMANY NORTHEAST
COMPUTE								
Virtual Machines		✓	✓	✓	✓		✓	✓
SAP HANA on Azure Large Instances		✓	✓					
Cloud Services		✓	✓	✓	✓		✓	✓

Resource Deployments Overview



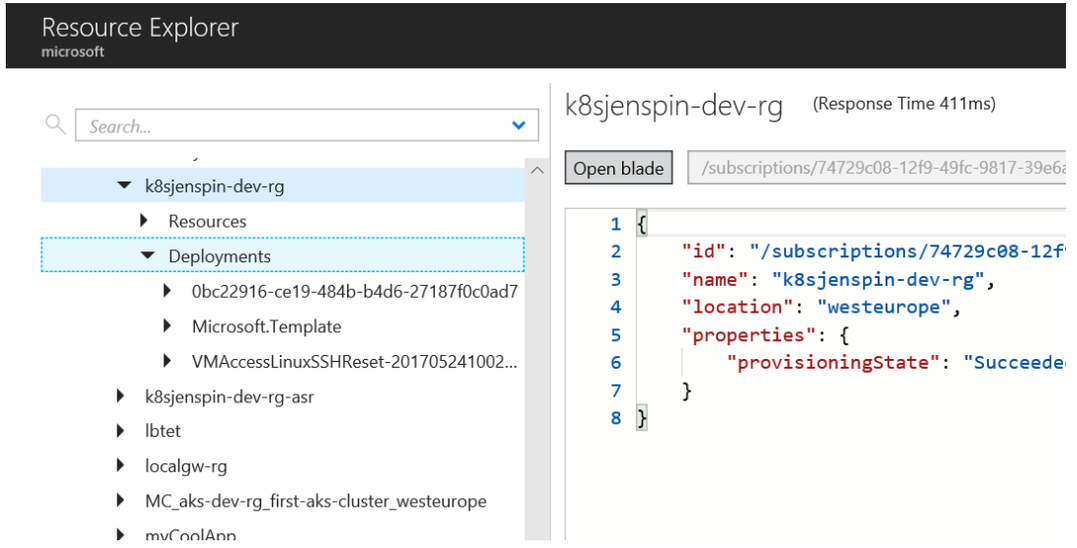
Resource Deployments

- Tools
  - CLI, PSH, Automation, Visual Studio, Logic App, VSTS
  - Tutorials: Azure PowerShell | Azure CLI | Azure portal | Resource Manager REST API

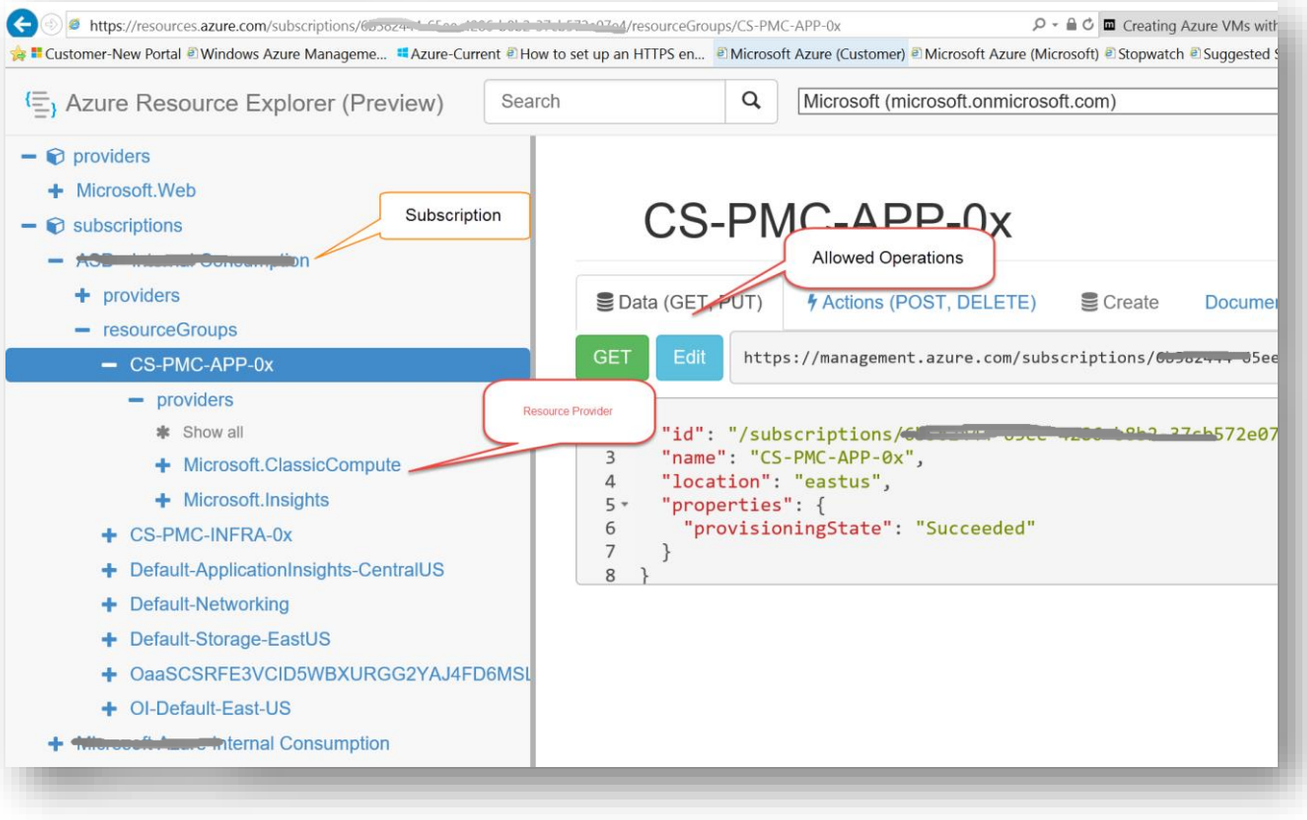
- Mode
  - **Complete** - ARM deletes resources that exist in the resource group but are not specified in the template. *Mode switch in PSH.*
  - **Incremental** (default) - ARM leaves unchanged resources that exist in the resource group but are not specified in the template.
  - More info

Resource Explorers

Azure Portal



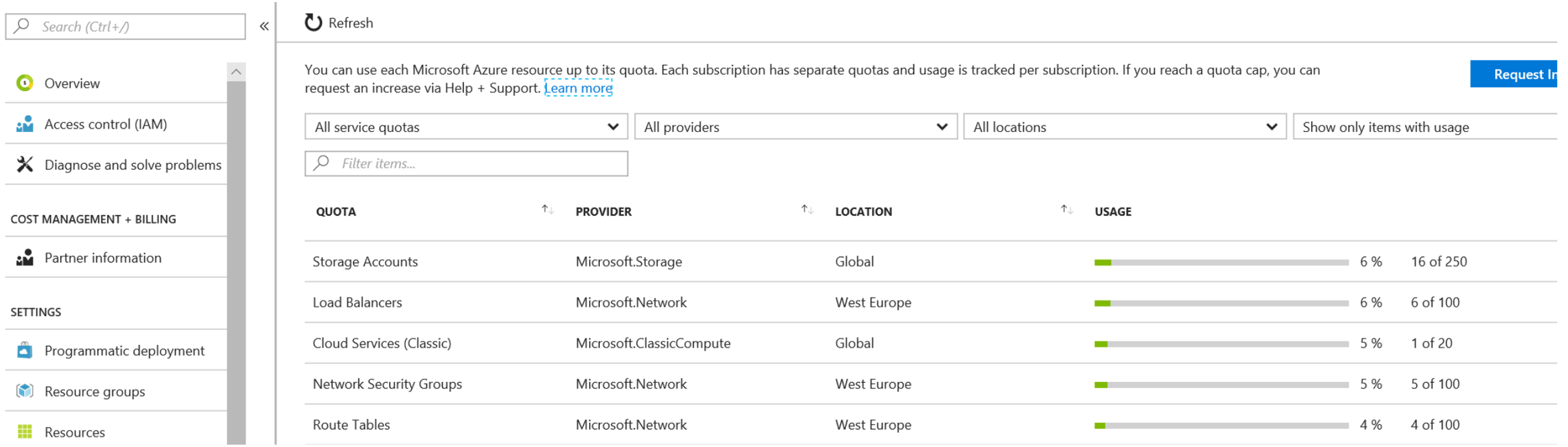
resources.azure.com



REST API Reference: https://docs.microsoft.com/en-us/rest/api/

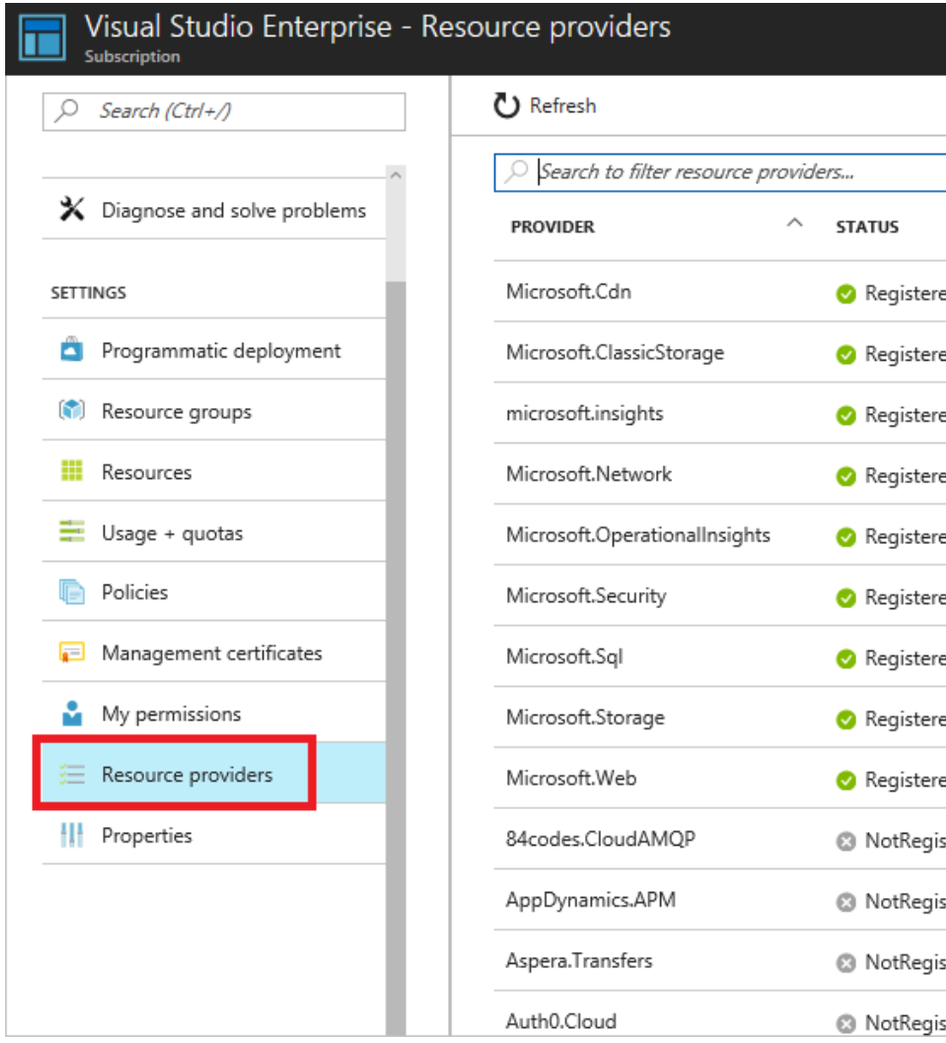
Resource limits

- Documentation: https://aka.ms/quotas/documentation
- Portal: Subscriptions – Usage + Quotas
- TIP: Get an alert programmatically – link (*Tom Hollander's blog*)



Resource Providers

- Registration of a Resource Provider
  - scope for registration is always the subscription
  - many RPs are automatically registered
  - manual registration – requires a permission to perform the **`/register/action`** operation for the resource provider. This operation is included in the Contributor and Owner roles.
  - You cannot unregister a resource provider when you still have resource types from that resource provider in your subscription.



Resource naming rules and restrictions

- Naming convention best practices: link
- Example:

Entity	Scope	Length	Casing	Valid Characters	Suggested Pattern	Example
Resource Group	Subscription	1-90	Case insensitive	Alphanumeric, underscore, parentheses, hyphen, and period (except at end)	`<service short name>-<environment>-rg`	`profx-prod-rg`
Availability Set	Resource Group	1-80	Case insensitive	Alphanumeric, underscore, and hyphen	`<service-short-name>-<context>-as`	`profx-sql-as`
Tag	Associated Entity	512 (name), 256 (value)	Case insensitive	Alphanumeric	`"key" : "value"`	`"department" : "Central IT"`

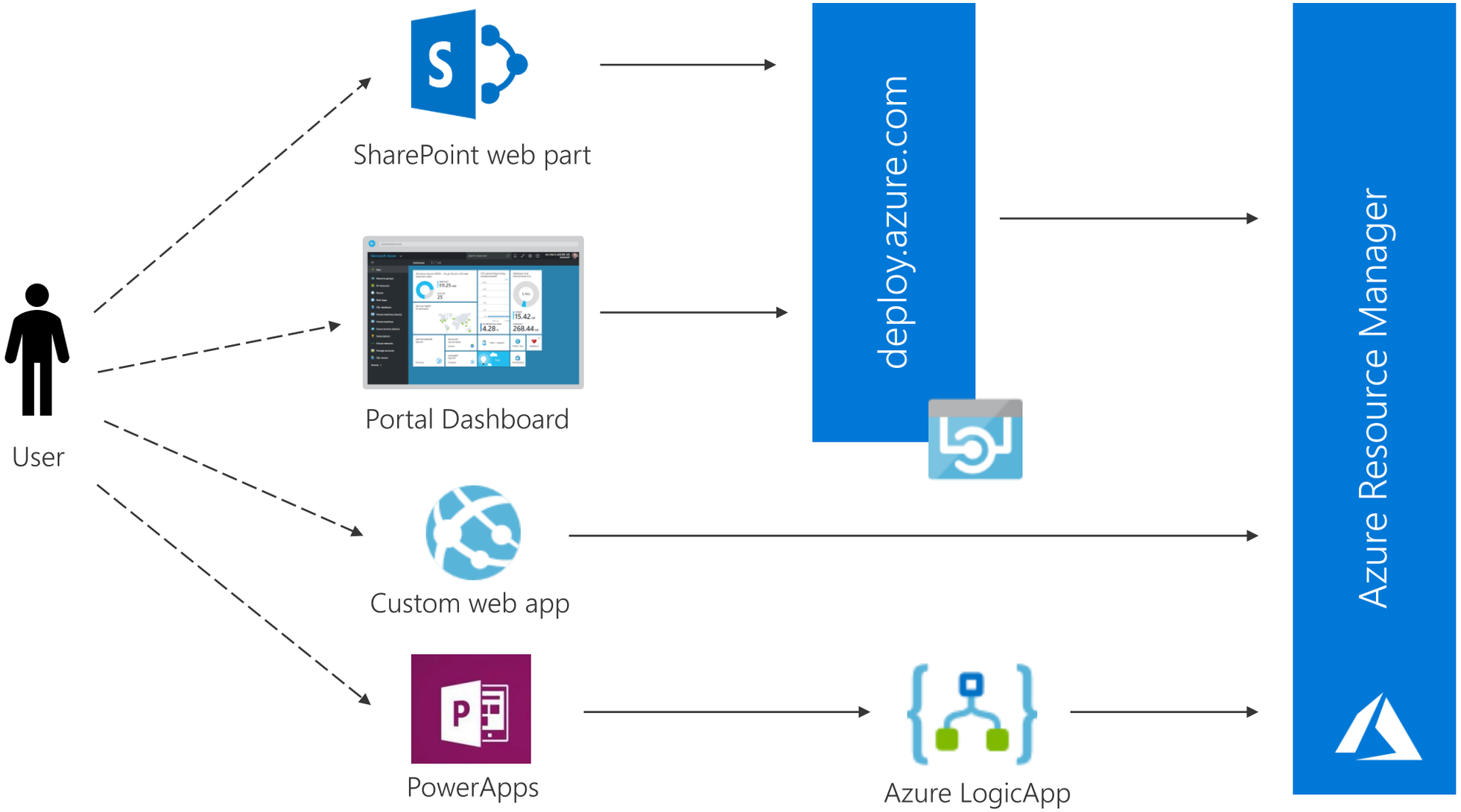
Episode 2: Douglas wants a catalogue

Building a simple Service Catalogue…

Scenario

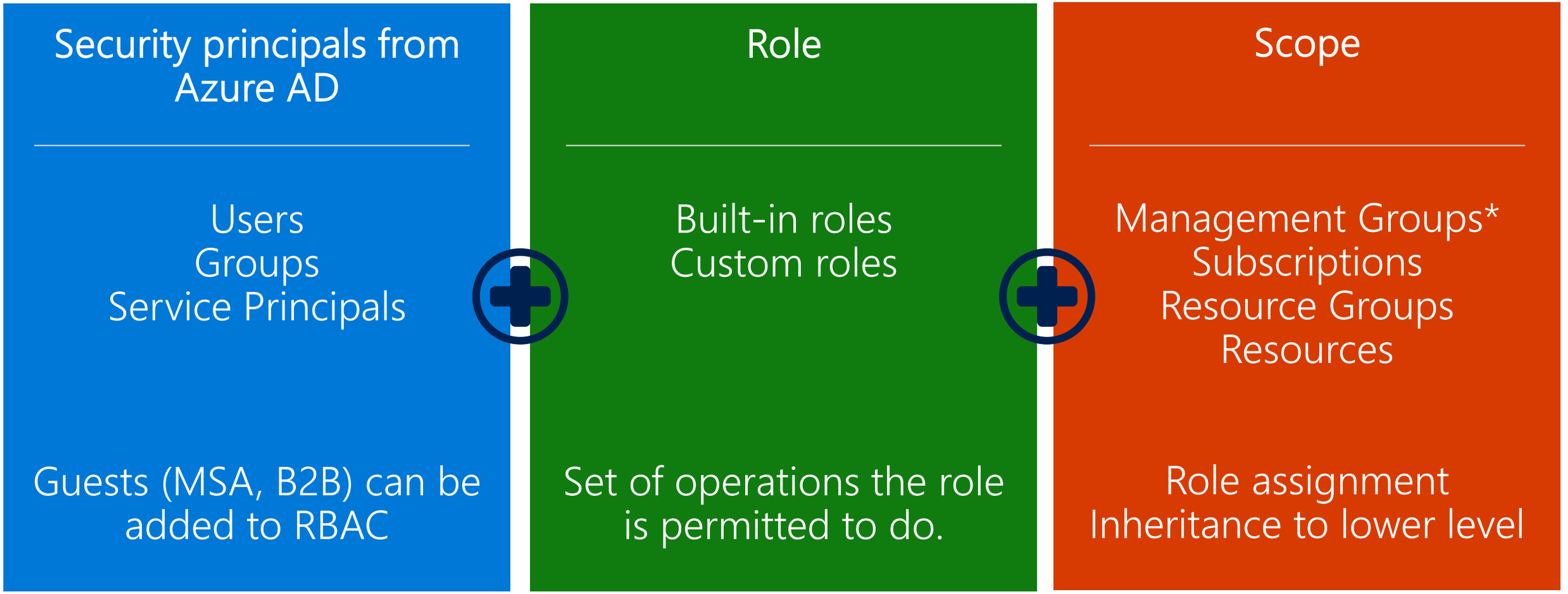
- Douglas wants Roy to build a service catalogue in Azure that would be extensible and easy to use.
- In this episode:
  - Extending Templates
  - Building the catalogue (outside of the Azure portal)
  - Marketplace solutions
  - T-shirt sizes
  - Sharing dashboards for users

Some options for building a catalogue



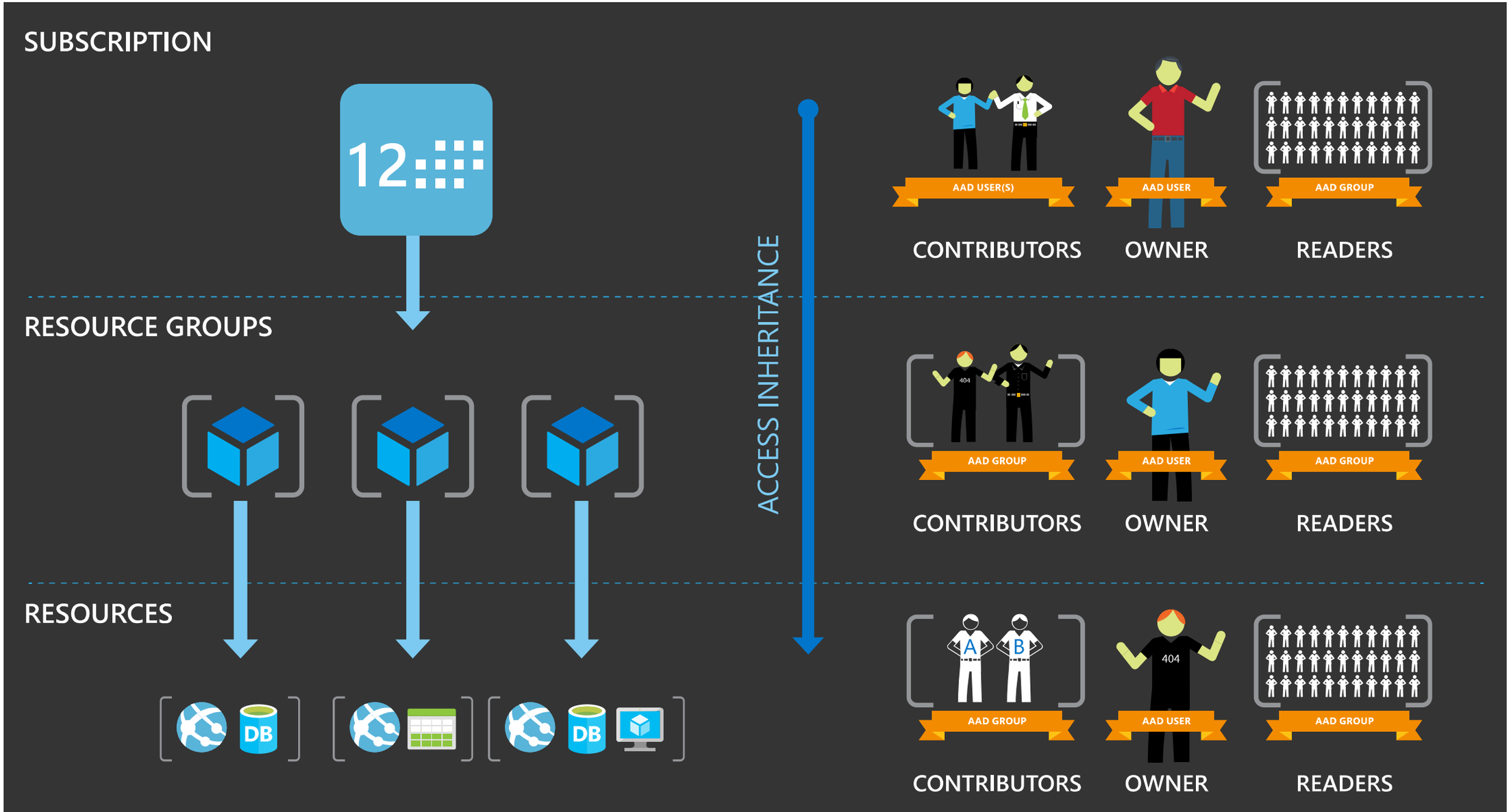
Production readiness and security hardening

RBAC model – define granular access control



** Currently in Preview*

Role-Based Access Control



Available roles

Built-in roles

BUILT-IN ROLE	ACTIONS	NOT ACTIONS
Owner (allow all actions)	*	
Contributor (allow all actions except writing or deleting role assignments)	*	Microsoft.Authorization/*/Write, Microsoft.Authorization/*/Delete
Reader (allow all read actions)	*/Read	

Custom roles

- Can be created using RBAC command-line tools in Azure PowerShell, and Azure Command-Line Interface (not in the Portal)

Two Key Concepts

Role Definitions

- describes the set of permissions (e.g. read actions)
- can be used in multiple assignments across your subscriptions

Role Assignments

- associate role definitions with an identity (e.g. user/group) at a scope (e.g. resource group)
- always inherited – subscription assignments apply to all resources

Custom role definition

- Defined in JSON
- Can be created in Azure PowerShell or CLI
- Custom Roles grant access to a list of actions
- Roles do not deny actions

```
{
  "Name": "Virtual Machine Operator",
  "Id": "cadb4a5a-4e7a-47be-84db-05cad13b6769",
  "IsCustom": true,
  "Description": "Can monitor and restart virtual machines.",
  "Actions": [
    "Microsoft.Storage/*/read",
    "Microsoft.Network/*/read",
    "Microsoft.Compute/*/read",
    "Microsoft.Compute/virtualMachines/start/action",
    "Microsoft.Compute/virtualMachines/restart/action",
    "Microsoft.Authorization/*/read",
    "Microsoft.Resources/subscriptions/resourceGroups/read",
    "Microsoft.Insights/alertRules/*",
    "Microsoft.Insights/diagnosticSettings/*",
    "Microsoft.Support/*"
  ],
  "NotActions": [

  ],
  "AssignableScopes": [
    "/subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e",
    "/subscriptions/e91d47c4-76f3-4271-a796-21b4ecfe3624",
    "/subscriptions/34370e90-ac4a-4bf9-821f-85eeedeae1a2"
  ]
}
```

RBAC - Granular Scopes

/subscriptions/{id}/resourceGroups/{name}/providers/.../sites/{site}

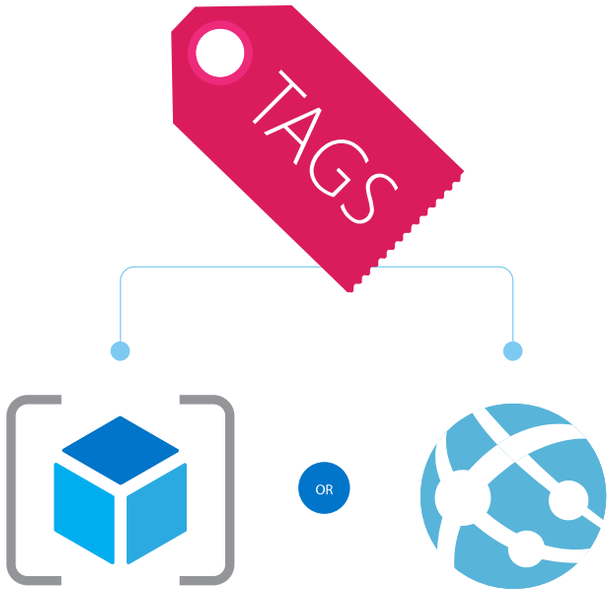
subscription level – grants permissions to all resources in the sub

resource group level – grants permissions to all resources in the group

resource level – grants permissions to the specific resource

Resource Tags – organize resources

- **Name-value pairs** assigned to resources or resource groups
- Subscription-wide taxonomy
- Each resource can have **up to 15 tags**
- Search/view resources with Tags
- Tags can be added to **ARM templates** and can be enforced with **policies**
- Usage: resource and cost management



Tags in a template

```
"resources": [
{
   "type": "Microsoft.Compute/virtualMachines",
   "apiVersion": "2015-06-15",
   "name": "SimpleWindowsVM",
   "location": "[resourceGroup().location]",
   "tags": {
      "costCenter": "Finance"
      },
 ...
}
]
```

Policies – ensuring compliance

- Enforce different **rules and actions over your resources** → compliance with your standards and SLAs
- Focuses on resource properties during deployment and for already existing resources - **evaluation**
- Examples: require specific tags, restrict certain resources, control locations, allow VM SKUs or storage SKUs, etc.
- Unlike RBAC, policy is a default allow and explicit deny system.
- *Contributor does not have permissions to define or assign policies.*

Policy characteristics

- Policy definition | Initiative definition
- Policy assignment
- Built-in (35) vs. custom policies
- Policy / Initiative parameters
  - Example for ‘Mandatory Tags’ policy – one definition, three assignments with a different ‘Environment’ key (parameter) for different subs.
- New experience in the Portal (previously just JSON and PSH/CLI, later simple GUI, now full experience /w Exceptions and two Tiers + Compliance view)

Policy definition structure

- Defined in JSON
- Evaluate fields and audit, deny, or extend deployments
- Conditions: Equals, Like, Contains, In, ContainsKey
- Logical Operators: Not, And, Or
- Fields: name, kind, type, location, tags, tags.*.
- More info

```
{
  "properties": {
    "mode": "all",
    "parameters": {
      "allowedLocations": {
        "type": "array",
        "metadata": {
          "description": "The list of locations that can be specified wh
          "strongType": "location",
          "displayName": "Allowed locations"
        }
      }
    },
    "displayName": "Allowed locations",
    "description": "This policy enables you to restrict the locations yo
    "policyRule": {
      "if": {
        "not": {
          "field": "location",
          "in": "[parameters('allowedLocations')]"
        }
      },
      "then": {
        "effect": "deny"
      }
    }
  }
}
```

Resource Locks

- Ability to lock a subscription, RG, or resource to prevent other users in your organization from **accidentally deleting or modifying critical resources**
  - lock levels: CanNotDelete or ReadOnly
  - Inheritance from a parent scope
- apply a restriction across all users and roles
- only Owner and User Access Admins can apply / modify them
- apply only to operations that happen in the management plane (management.azure.com)
- More info

Episode 3: New CISO arrives

Adding some security and compliance stuff…

Scenario

- Richmond got promoted to a CISO role and defined a new security policy for Reynholm Industries.
- In this episode:
  - Least privilege access and custom roles
  - Limiting human errors in production with locks
  - Standardizing the environment with resource metadata
  - Checking compliance

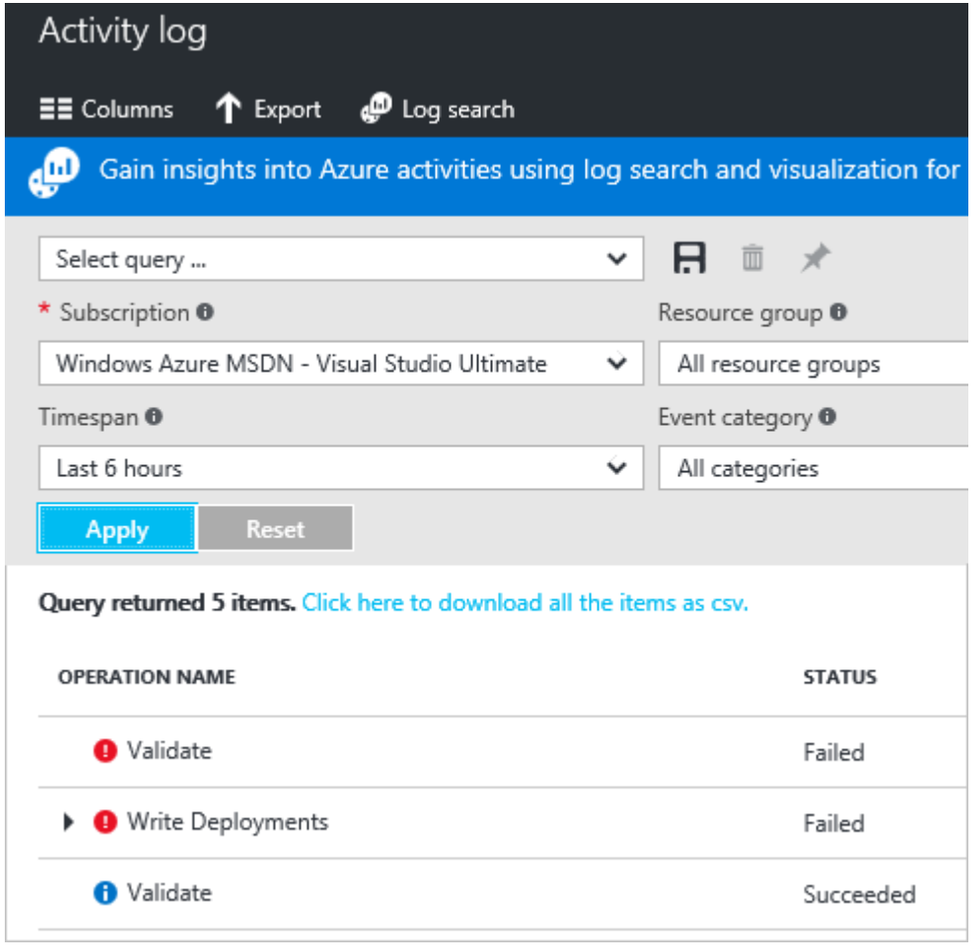
Reynholm Ind. – IT Security Policy for Azure

- Least privilege for all users – limit number of subscription Owners
- Secure production resources from accidental deletion and limiting who could remove locks
- Apply mandatory tags for better resource and cost management (Environment, CostPool)
- Production data stores cannot be deployed outside of EU. Compliance with this policy shall not be enforced first but it must be checked regularly.

Building a robust and automated delivery

Deployment audit

- Deployment error types:
  - **validation errors** - syntax errors in your template, or trying to deploy resources that would exceed your subscription quotas
  - **deployment errors** - arise from conditions that occur during the deployment process
- Activity Log
- Alerts
- More info



Verbose mode in PowerShell

```
New-AzureRmResourceGroupDeployment -Name test1 -ResourceGroupName $resourceGroupName -
TemplateFile $templateFile -TemplateParameterFile $templatePrameterFile -Verbose
```

```
PS C:\Windows\System32\WindowsPowerShell\v1.0> C:\Users\kay\Desktop\templaterun.ps1


ResourceGroupName : kayrg2
Location          : westus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/[redacted]/resourceGroups/kayrg2

VERBOSE: 9:42:17 PM - Create template deployment 'test1'.
VERBOSE: 9:42:23 PM - Resource Microsoft.Network/publicIPAddresses 'myPublicIP' provisioning status is running
VERBOSE: 9:42:23 PM - Resource Microsoft.Storage/storageAccounts 'xvoxcf6yywftosalinuxvm' provisioning status is running
VERBOSE: 9:42:23 PM - Resource Microsoft.Network/virtualNetworks 'MyVNET' provisioning status is running
VERBOSE: 9:42:34 PM - Resource Microsoft.Network/publicIPAddresses 'myPublicIP' provisioning status is succeeded
VERBOSE: 9:42:34 PM - Resource Microsoft.Network/virtualNetworks 'MyVNET' provisioning status is succeeded
VERBOSE: 9:42:41 PM - Resource Microsoft.Network/networkInterfaces 'myVMNic' provisioning status is succeeded
VERBOSE: 9:42:50 PM - Resource Microsoft.Storage/storageAccounts 'xvoxcf6yywftosalinuxvm' provisioning status is succeeded
VERBOSE: 9:42:54 PM - Resource Microsoft.Compute/virtualMachines 'MyUbuntuVM' provisioning status is running
```

Debugging ARM Deployments

- Review Activity log
  - Journals all write/delete/actions; log retention is 90 days
- Review Deployment log
- Refresh expired credentials in PowerShell or Visual Studio
- Check format of Templates
- Check Location supports resource
- Check valid and available names
- Check for access rules
- Check for quota limits

Inside the Box vs. Outside the Box

- Template describes the **topology** (outside the box)
- Template **extensions can initiate state configuration** (inside the box)
- Multiple extensions available
  - DSC
  - Chef
  - Puppet
  - Custom Scripts
  - AppService + WebDeploy
  - SQLDB + BACPAC

Template language expressions*

```
base64encode(‘stringtoencode’)
concat(‘string’,’to’,’encode’)
copyIndex(offset)
listKeys(storageAccountResourceId, apiVersion)
padLeft(stringToPad,targetLength,paddingCharacter)
parameters(‘parameterName’)
providers(namespace, resourceType)
reference(resourceId,apiVersion)
resourceGroup()
resourceId(‘namespace/resourceType', ‘resourceName’)
subscription()
variables(‘variables’)
```

*Looking for examples? See these in action @ https://github.com/rjmax/ArmExamples

Most common

```
parameters('parameterName')
variables('variableName')
concat('string', 'to', 'join')
```

Usage

```
"subnetId": "[concat(variables('vnetID'), '/subnets/', variables('subnetName'))]"
```

Other functions

```
resourceId('namespace/resourceType', 'resourceName')
listKeys('storageAccountResourceId', 'apiVersion')
uniqueString ('stringForCreatingUniqueString', ...)
toLower('stringToChange')
substring('stringToParse', startIndex, length)
```

Complete list available at
https://azure.microsoft.com/en-in/documentation/articles/resource-group-template-functions/

Consistency is key

- Use **parameters** for resource names that need to be set on the fly
- Use **variables** or **hard-coded** resource names otherwise
- camelCasing is recommended
- use “metadata” to add descriptions

```
"parameters": {
    "storageAccountType": {
        "type": "string",
        "defaultValue": "Standard_GRS",
        "metadata": {
            "description": "The type of the new storage account created to store the VM disks"
        }
    }
}
```

Nesting Templates

```
"resources": [
  {
     "apiVersion": "2015-01-01",
     "name": "nestedTemplate",
     "type": "Microsoft.Resources/deployments",
     "properties": {
       "mode": "incremental",
       "templateLink": {
          "uri": "https://www.contoso.com/AzureTemplates/newStorageAccount.json",
          "contentVersion": "1.0.0.0"
       },
       "parameters": {
          "StorageAccountName":{"value": "[parameters('StorageAccountName')]"}
       }
     }
  }
]
```

- TemplateLink must be visible to Resource Manager: http or https uri
- Can use variables to create multiple URLs from base name

Deploying Multiple Instances

- Resource loops deploy n instances
- Fixed or parameter driving instance count
- Concat + Parameter Prefix + CopyIndex() for dynamic naming

```
{
  "apiVersion": "2015-05-01-preview",
  "type": "Microsoft.Compute/virtualMachines",
  "name": "[concat(parameters('vmNamePrefix'), copyindex())]",
  "location": "[parameters('location')]",
  "copy": {
    "name": "virtualMachineLoop",
    "count": "[parameters('numberOfInstances')]"
  },
  "dependsOn": [
    "[concat('Microsoft.Network/networkInterfaces/', 'nic', copyindex())]"
  ],
  "properties": {
    "hardwareProfile": {
      "vmSize": "[parameters('vmSize')]"
    },
    "osProfile": {
      "computername": "[concat('vm', copyIndex())]",
      "adminUsername": "[parameters('adminUsername')]",
      "adminPassword": "[parameters('adminPassword')]"
    },
    "storageProfile": {
      "osDisk": {
        "name": "[concat(parameters('vmNamePrefix'),'-osDisk',copyindex())]",
        "osType": "[parameters('osType')]",
        "caching": "ReadWrite",
        "image": {
          "uri": "[variables('userImageName')]"
        },
        "vhd": {
          "uri":
"[concat(variables('osDiskVhdContainer'),parameters('vmNamePrefix'),copyindex(),'osDisk.vhd')]"
        }
      }
    },
```

Passing State – Output variables

- A template can return values to its caller via the outputs section

```
"outputs": {
"masterip": {
"value":
"[reference(concat(variables('nicName'),0)).ipConfigurations[0].properties.privateIPAddre
ss]",
"type":"string"
}}
```

- These values can then be used by the caller

```
"masterIpAddress": {
"value":
"[reference('master-node').outputs.masterip.value]"
 } }
```

Passing State - Common Parameters

Name	Value	Description
region	String from a constrained list of Azure regions	The location where the resources will be deployed.
storageAccountNamePrefix	String	Unique DNS name for the Storage Account where the VM's disks will be placed
domainName	String	Domain name of the publicly accessible jumpbox VM in the format: `{domainName}.{location}.cloudapp.com` For example: `mydomainname.westus.cloudapp.azure.com`
adminUsername	String	Username for the VMs
adminPassword	String	Password for the VMs
tshirtSize	String from a constrained list of offered t-shirt sizes	The named scale unit size to provision. For example, "Small", "Medium", "Large"
virtualNetworkName	String	Name of the virtual network that the consumer wants to use.
jumpbox	String from a constrained list (enabled/disabled)	Parameter that identifies whether to enable a jumpbox for the environment. Values: "enabled", "disabled"

Control Flow

- No control flow logic built into ARM template language
- An approach with parameters, variables, and linked templates
  - Use provides parameter value that provides context, e.g. tshirtSize parameter is passed in as a value of ‘small’
  - Using concat and a pre-defined variable, a new variable value is created which points to the specific , e.g. ‘tshirtSize-small.json’
  - Template linking is incorporated into the template and uses this new value to identify which template to deploy.
  - Common examples are “tshirt sizes” and optional features for a deployment, e.g. “enableJumpbox”

Best practices - security

- Use Azure Key Vault with Resource Manager to orchestrate and store VM secrets and certificates
- Separate keys from deployments
  - Template 1: Creation of vaults (which will contain the key material)
  - Template 2: Deployment of the VMs (with URI references to the keys contained in the vaults)
- Use AD service principals for cross-subscription interactions
- Use Network Security Groups to control traffic to VMs in a Virtual Network

Other best practices

- API version should be hardcoded
- Location should be set with resourceGroup().location when possible
- Use the securestring or Azure Key Vault for all passwords and secrets
- Minimize parameters when possible
- Take advantage of the comments element
- Always use reference() and resourceId() over hardcoded endpoints and IDs
- Output any newly created endpoints (e.g. public IPs)
- Consider using a JSON validator (for example in the Azure Portal) and automated formatting for better readability

Episode 4: Automated Moss

Automating deployments and IT processes…

Scenario

- Moss wants to go “full monty” to automate deployments of new apps, and remove secrets from deployments. Roy wants to automate a few IT processes and tackle some complex scenarios.
- In this episode:
  - Deployment audit and alerting
  - Nested templates with functions
  - Removing secrets from VMs
  - Automated checks for newly deployed VMs

Resources

Resources

- Courses
  - Infrastructure as Code (MVA)
  - Deploying production workloads with Azure Resource Manager (MVA)
  - Mastering Microsoft Azure Resource Manager (Pluralsight)
  - Automating Deployment and Scale of Azure IaaS Solutions (Pluralsight)
- Learning Portals
  - Azure Training (azure.com)
    - Free Pluralsight courses | Learning paths | Hands-on labs | Certifications
  - Azure Essentials (link)

Resources

- Documents
  - World Class ARM Templates – Considerations and Proven Practices (PDF)

Tips

- Azure Interactives
  - http://azureinteractives.azurewebsites.net/
  - Three interactive experiences: Azure Products, Cloud Design Patterns, Azure Security + Operations Management
- Azure Cloud Shell
  - https://shell.azure.com
  - Bash + PowerShell
  - CloudDrive
  - Tools: k8s (kubectl), docker, git, terraform, ++

Microsoft